Learning Low-rank Deep Neural Networks via Singular Vector Orthogonality Regularization and Singular Value Sparsification

Huanrui Yang[1], Minxue Tang[2], Wei Wen[1], Feng Yan[3], Daniel Hu[4], Ang Li[1], Hai Li[1], Yiran Chen[1]
[1]Department of Electrical and Computer Engineering, Duke University
[2]Department of Electronic Engineering, Tsinghua University
[3]Computer Science and Engineering Department, University of Nevada, Reno
[4]Newport High School, Bellevue, WA
[1]{huanrui.yang, wei.wen, ang.li630, hai.li, yiran.chen}@duke.edu, [2]tangmx16@mails.tsinghua.edu.cn,
[3]fyan@unr.edu, [4]danielhu2003@gmail.com

Abstract

Modern deep neural networks (DNNs) often require high memory consumption and large computational loads. In order to deploy DNN algorithms efficiently on edge or mobile devices, a series of DNN compression algorithms have been explored, including factorization methods. Factorization methods approximate the weight matrix of a DNN layer with the multiplication of two or multiple low-rank matrices. However, it is hard to measure the ranks of DNN layers during the training process. Previous works mainly induce low-rank through implicit approximations or via costly singular value decomposition (SVD) process on every training step. The former approach usually induces a high accuracy loss while the latter has a low efficiency. In this work, we propose ***SVD training****, the first method to explicitly achieve low-rank DNNs during training without applying SVD on every step. SVD training first decomposes each layer into the form of its full-rank SVD, then performs training directly on the decomposed weights. We add orthogonality regularization to the singular vectors, which ensure the valid form of SVD and avoid gradient vanishing/exploding. Low-rank is encouraged by applying sparsity-inducing regularizers on the singular values of each layer. Singular value pruning is applied at the end to explicitly reach a low-rank model. We empirically show that SVD training can significantly reduce the rank of DNN layers and achieve higher reduction on computation load under the same accuracy, comparing to not only previous factorization methods but also state-of-the-art filter pruning methods.*

1. Introduction

The booming development in deep learning models and applications has enabled beyond human performance in tasks like large-scale image classification [18, 9, 12, 13], object detection [27, 22, 8], and semantic segmentation [24, 3]. Such high performance, however, comes with a high price of large memory consumption and computation load. For example, a ResNet-50 model needs approximately 4G floating-point operations (FLOPs) to classify a color image of 224×224 pixels. The computation load can easily expand to tens or even hundreds of GFLOPs for detection or segmentation models using state-of-the-art DNNs as backbones [2]. This is a major challenge that prevents the deployment of modern DNN models on resource-constrained platforms, such as phones, smart sensors, and drones.

Model compression techniques for DNN models, including element-wise pruning [7, 21, 38], structural pruning [33, 25, 20], quantization [23, 31], and factorization [15, 39, 36, 35], have been extensively studied. Among these methods, quantization and element-wise pruning can effectively reduce model's memory consumption, but require specific hardware to realize efficient computation. Structural pruning reduces the computation load by removing redundant filters or channels. However, the complicated structures adopted in some modern DNNs (i.e., ResNet or DenseNet) enforce strict constraints on the input/output dimensions of certain layers. This requires additional filter grouping during the pruning and filter rearranging after the pruning to make the pruned structure valid [32, 5]. Factorization method approximates the weight matrix of a layer with a multiplication of two or more low-rank matrices. It by nature keeps the input/output dimension of a layer unchanged, and therefore the resulted decomposed network can be supported by any common DNN computation architectures, without additional grouping and post-processing.

The previous investigation show that it is feasible to approximate the weight matrices of a pretrained DNN model with the multiplication of low-rank matrices [15, 39, 26,

4, 19]. But these methods may greatly degrade the performance, even after post-hoc finetuning. Some other methods attempt to manipulate the "directions" of filters to implicitly reduce the rank of weight matrices [34, 20]. However, the difficulties in training and the implicitness of rank representation prevent these methods from reaching a high compression rate. Nuclear norm regularizer has been used to directly reduce the rank of weight matrices [1, 35]. Optimizing the nuclear norm requires conducting singular value decomposition (SVD) on every training step, which is inefficient, especially when dealing with larger models.

Our work aims to explicitly achieve a low-rank DNN network during the training without applying SVD on every step. In particular, we propose *SVD training* by training the weight matrix of each layer in the form of its full-rank SVD. The weight matrix is decomposed into the matrices of left-singular vectors, singular values and right-singular vectors, and the training is done on the decomposed variables. Furthermore, two techniques are proposed to induce low-rank while maintaining high performance during the SVD training: (1) *Singular vector orthogonality regularization* which keeps the singular vector matrices close to unitary throughout the training. It mitigates gradient vanishing/exploding during the training, and provide a valid form of SVD to guarantee the effective rank reduction. (2) *Singular value sparsification* which applies sparsity-inducing regularizers on the singular values during the training to induce low-rank. The low-rank model is finally achieved through singular value pruning. We evaluate the individual contribution of each technique as well as the overall performance when putting them together via ablation studies. Results show that the proposed method constantly beats state of the art factorization and structural pruning methods on various tasks and model structures. To the best of our knowledge, this is the first algorithm to explicitly search for the optimal rank of each DNN layer during the training without performing the decomposition operation at each training step.

2. Related Works on low-rank DNNs

Approximating a weight matrix with the multiplication of low-rank matrices is a straightforward idea for compressing DNNs. Early works in this field focus on designing the matrix and tensor decomposition scheme, especially for the 4-D tensor of convolution kernel, so that the operation of a pretrained network layer can be closely approximated with cascaded low-rank layers [15, 39, 26, 4, 19]. Tensor decomposition technique, notably CP-decomposition, is applied in early works to directly decompose the 4-D convolution kernel into 4 consecutive low-rank convolutions [19]. However, such decomposition technique significantly increases the number of layers in the achieved network, making them harder to be finetuned towards good performance, especially when decomposing larger and deeper models [29]. Later works therefore propose to reshape the 4-D tensor into a 2-D matrix, apply matrix decomposition technique like SVD to decompose the matrix, and finally reshape them back to 4-D tensors to get two consecutive layers. Notably, Zhang et al. [39] propose channel-wise decomposition, which uses SVD to decompose a convolution layer with a kernel size $w \times h$ into two consecutive layers with kernel sizes $w \times h$ and 1×1, respectively. The computation reduction can be achieved by exploiting the channel-wise redundancy, e.g., channels with smaller singular values in both decomposed layers are removed. Similarly, Jaderberg et al. [15] propose to decompose a convolution layer into two consecutive layers with less channels in between. They further utilize the spatial-wise redundancy to reduce the size of convolution kernels in the decomposed layers to $1 \times h$ and $w \times 1$, respectively. These methods provide a closed-form decomposition for each layer. If the SVD is done in full rank, these methods guarantee that the decomposed layers perform the same operation as the original layer. However, the weights of the pretrained model may not be low-rank by nature, so the manually imposed low-rank after decomposition by removing small singular values inevitably leads to high accuracy loss as the compression ratio increases [35].

Methods have been proposed to reduce the ranks of weight matrices during training process in order to achieve low-rank decomposition with low accuracy loss. Wen et al. [34] induce low rank by applying an "attractive force" regularizer to increase the correlation of different filters in a certain layer. Ding et al. [5] achieve a similar goal by optimizing with "centripetal SGD," which moves multiple filters towards a set of clustering centers. Both methods can reduce the rank of the weight matrices without performing actual low-rank decomposition during the training. However, the rank representations in these methods are implicit, so the regularization effects are weak and may lead to sharp performance decrease when seeking for a high speedup. On the other hand, Alvarez et al. [1] and Xu et al. [35] explicitly estimate and reduce the rank throughout the training by adding Nuclear Norm (defined as the sum of all singular values) regularizer to the training objective. These method require performing SVD to compute and optimize the Nuclear Norm of each layer on every optimization step. Since the complexity of the SVD operation is $\mathcal{O}(n^3)$ and the gradient computation through SVD is not straightforward [6], performing SVD on every step is time consuming.

To explicitly achieve a low-rank network without performing costly decomposition on each training step, Tai et al. [29] propose to directly train the network from scratch in the low-rank decomposed form, and add batch normalization [14] between decomposed layers to tackle the potential gradient vanishing or exploding problem caused by the doubling of layers after decomposition. However, the low-rank decomposed training scheme used in this line of works re-

quires setting the rank of each layer before the training [29]. The manually chosen low rank may not lead to the optimal compression. Also, training the low-rank model from scratch will make the optimization harder, as lower rank implies lower model capacity [35].

3. Proposed Method

Building upon previous works, we combine the ideas of decomposed training and trained low-rank in this work. As shown in Figure 1, the model will first be trained in a decomposed form through the full-rank SVD training, then undergoes singular value pruning for rank reduction, and finally be finetuned for further accuracy recovery. As we will explain in Section 3.1, the model will be trained in the form of the spatial-wise [15] or channel-wise decomposition [39] to avoid the time consuming SVD. Unlike the training procedure proposed by [29], we will train the decomposed model in its full-rank to preserve the model capacity. During the SVD training, we apply orthogonality regularization to the singular vector matrices and sparsity-inducing regularizers to the singular values of each layer, the details of which will be discussed in Section 3.2 and 3.3, respectively. Section 3.4 will elaborate the full objective of the SVD training and the overall model compression pipeline. This method is able to achieve optimal compression rate by inducing low-rank through training without the need for performing decomposition on every training step.

3.1. SVD training of deep neural networks

In this work, we propose to train the neural network in its singular value decomposition form, where each layer is decomposed into two consecutive layers with no additional operations in between. For a fully connected layer, the weight $\boldsymbol{W}$ is a 2-D matrix with dimension $\boldsymbol{W} \in \mathbb{R}^{m\times n}$. Following the form of SVD, $\boldsymbol{W}$ can be directly decomposed into three variables $\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{s}$ as $\boldsymbol{U}\text{diag}(\boldsymbol{s})\boldsymbol{V}^T$, with dimension $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ and $\boldsymbol{s} \in \mathbb{R}^r$. Both $\boldsymbol{U}$ and $\boldsymbol{V}$ shall be unitary matrices. In the full-rank setting where $r = \min(m, n)$, $\boldsymbol{W}$ can be exactly reconstructed as $\boldsymbol{W} = \boldsymbol{U}\text{diag}(\boldsymbol{s})\boldsymbol{V}^T$. For a neural network, this is equivalent to decomposing a layer with weight $\boldsymbol{W}$ into two consecutive layers with weight $\boldsymbol{W}_1 = \boldsymbol{U}\text{diag}(\sqrt{\boldsymbol{s}})$ and $\boldsymbol{W}_2 = \text{diag}(\sqrt{\boldsymbol{s}})\boldsymbol{V}^T$ respectively.

For a convolution layer, the kernel $\mathbf{K}$ can be represented as a 4-D tensor with dimension $\mathbf{K} \in \mathbb{R}^{n\times c\times w\times h}$. Here n, c, w, h represent the numbers of filters, the number of input channels, the width and the height of the filter respectively. This work mainly focuses on the *channel-wise decomposition* method [39] and the *spatial-wise decomposition* method [15] to decompose the convolution layer, as these methods have shown their effectiveness in previous CNN decomposition research. For channel-wise decomposition, $\mathbf{K}$ is first reshaped to a 2-D matrix $\hat{\boldsymbol{K}} \in \mathbb{R}^{n\times cwh}$. $\hat{\boldsymbol{K}}$ is then decomposed with SVD into $\boldsymbol{U} \in \mathbb{R}^{n\times r}$, $\boldsymbol{V} \in \mathbb{R}^{cwh\times r}$ and $\boldsymbol{s} \in \mathbb{R}^r$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices and $r = \min(n, cwh)$. The original convolution layer is therefore decomposed into two consecutive layers with kernels $\mathbf{K}_1 \in \mathbb{R}^{r\times c\times w\times h}$ reshaped from $\text{diag}(\sqrt{\boldsymbol{s}})\boldsymbol{V}^T$ and $\mathbf{K}_2 \in \mathbb{R}^{n\times r\times 1\times 1}$ reshaped from $\boldsymbol{U}\text{diag}(\sqrt{\boldsymbol{s}})$. Spatial-wise decomposition shares a similar process as the channel-wise decomposition. The major difference is that $\mathbf{K}$ is now reshaped to $\hat{\boldsymbol{K}} \in \mathbb{R}^{nw\times ch}$ and then decomposed into $\boldsymbol{U} \in \mathbb{R}^{nw\times r}$, $\boldsymbol{V} \in \mathbb{R}^{ch\times r}$, and $\boldsymbol{s} \in \mathbb{R}^r$ with $r = \min(nw, ch)$. The resulting decomposed layers would have kernels $\mathbf{K}_1 \in \mathbb{R}^{r\times c\times 1\times h}$ and $\mathbf{K}_2 \in \mathbb{R}^{n\times r\times w\times 1}$ respectively. [39] and [15] theoretically show that the decomposed layers can exactly replicate the function of the original convolution layer in the full-rank setting. Therefore training the decomposed model at full-rank should achieve a similar accuracy as training the original model.

During the SVD training, for each layer we use the variables from the decomposition, i.e., $\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}$, instead of the original kernel $\mathbf{K}$ or weight $\boldsymbol{W}$ as the trainable variables in the network. The forward pass will be executed by converting the $\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}$ into a form of the two consecutive layers as demonstrated above, and the back propagation and optimization will be done directly with respect to the $\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}$ of each layer. In this way, we can access the singular value $\boldsymbol{s}$ directly without performing the time-consuming SVD on each step.

Note that $\boldsymbol{U}$ and $\boldsymbol{V}$ need to be orthogonal to guarantee the low rank approximation can be done by removing small singular values, but this is not naturally induced by the decomposed training process. Therefore we add orthogonality regularization to $\boldsymbol{U}$ and $\boldsymbol{V}$ to tackle this problem as discussed in Section 3.2. Rank reduction is induced by adding sparsity-inducing regularizers to the $\boldsymbol{s}$ of each layer, which will be discussed in Section 3.3.

3.2. Singular vectors orthogonality regularizer

In a standard SVD procedure, the resulted $\boldsymbol{U}$ and $\boldsymbol{V}$ should be orthogonal by construction, which provides theoretical guarantee for the low-rank approximation. However, $\boldsymbol{U}$ and $\boldsymbol{V}$ in each layer are treated as free trainable variables in the decomposed training process, so the orthogonality may not hold. Without the orthogonal property, it is unsafe to prune the singular value in $\boldsymbol{s}$ even if it reaches a small value, because the corresponding singular vectors in $\boldsymbol{U}$ and $\boldsymbol{V}$ may have high energy and lead to a large difference to the result.

To make the form of SVD valid and enable effective rank reduction via singular value pruning, we introduce an orthogonality regularization loss to $\boldsymbol{U}$ and $\boldsymbol{V}$ as:

$$L_o(\boldsymbol{U}, \boldsymbol{V}) = \frac{1}{r^2}(||\boldsymbol{U}^T\boldsymbol{U} - \boldsymbol{I}||_F^2 + ||\boldsymbol{V}^T\boldsymbol{V} - \boldsymbol{I}||_F^2), \quad (1)$$

where $||\cdot||_F$ is the Frobenius norm of matrix and r is the

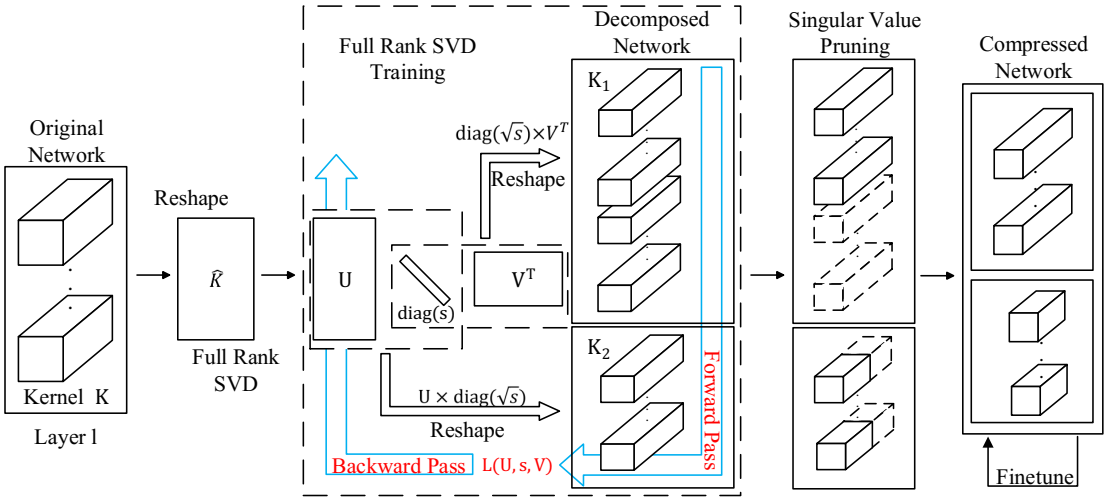

Figure 1. The training, compressing and finetuning pipeline of the proposed method.

rank of $\boldsymbol{U}$ and $\boldsymbol{V}$. Note that the ranks of $\boldsymbol{U}$ and $\boldsymbol{V}$ are the same given their definition in the decomposed training procedure. Adding the orthogonality loss in Equation (1) to the total loss function forces $\boldsymbol{U}$s and $\boldsymbol{V}$s of all the layers close to be orthogonal matrices.

Beyond maintaining valid SVD form, the orthogonality regularization also bring additional benefit to the performance of the decomposed training process. The decomposed training process convert one layer in the original network to two consecutive layers, therefore doubles the number of layers. As aforementioned in [29], this may worsen the problem of exploding or vanishing gradient during the optimization, degrading the performance of the achieved model. Since the proposed orthogonality loss can keep all the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ to have the L^2 norms close to 1, it can effectively prevent the gradient to explode or vanish when passing through variable $\boldsymbol{U}$ and $\boldsymbol{V}$, therefore helping the training process to achieve a higher accuracy. The accuracy gain brought by training with the orthogonality loss will be discussed in our ablation study in Section 4.1.

3.3. Singular values sparsity-inducing regularizer

With orthogonal singular vector matrices, reducing the rank of the decomposed network is equivalent to making the singular value vector $\boldsymbol{s}$ of each layer sparse. Although the sparsity of a vector is directly represented by its L^0 norm, it is hard to optimize the norm through gradient-based methods. Inspired by the recent works in DNN pruning [21, 33], we use differentiable sparsity-inducing regularizer to make more elements in $\boldsymbol{s}$ closer to zero, and apply post-train pruning to make the singular value vector sparse.

For the choice of the sparsity-inducing regularizer, the L^1 norm has been commonly applied in feature selection [30] and DNN pruning [33]. The L^1 regularizer takes the form of $L^1(\boldsymbol{s}) = \sum_i |s_i|$, which is both almost everywhere differentiable and convex, making it friendly for optimization. Moreover, applying L^1 regularizer on the singular value $\boldsymbol{s}$ is equivalent to regularizing with the nuclear norm of the original weight matrix, which is a popular approximation to the rank of a matrix [35].

However, the L^1 norm is proportional to the scaling of parameters, i.e., $||\alpha W||_1 = \alpha||W||_1$, with a non-negative constant α. Therefore, minimizing the L^1 norm of $\boldsymbol{s}$ will shrink all the singular values simultaneously. In such a situation, some singular values that are close to zero after training may still contain a large portion of the matrix's energy. Pruning such singular values may undermine the performance of the neural network.

To mitigate the proportional scaling problem of the L^1 regularizer, previous works in compressed sensing have been using Hoyer regularizer to induce sparsity in solving non-negative matrix factorization [11] and blind deconvolution [16], where the Hoyer regularizer shows superior performance comparing to other methods. The Hoyer regularizer is formulated as

$$L^H(\boldsymbol{s}) = \frac{||\boldsymbol{s}||_1}{||\boldsymbol{s}||_2} = \frac{\sum_i |s_i|}{\sqrt{\sum_i s_i^2}}, \tag{2}$$

which is the ratio of the L^1 norm and the L^2 norm of a vector [16]. It can be easily seen that the Hoyer regularizer is almost everywhere differentiable and scale-invariant. The differentiable property implies that the Hoyer regularizer can be easily optimized as part of the objective function. The scale-invariant property shows that if we apply the Hoyer regularizer to $\boldsymbol{s}$, the total energy will be retained as the singular values getting sparser. Therefore most of the energy will be kept within the top singular values while the rest getting close to zero. This makes Hoyer regularizer attractive in our training process. The effectiveness of the L^1 regularizer and the L^H regularizer is explored and compared in Section 4.3.

3.4. Overall objective and training procedure

With the analysis above, we propose the overall objective function of the decomposed training as:

$$L(\boldsymbol{U}, \boldsymbol{s}, \boldsymbol{V}) = L_T(\text{diag}(\sqrt{|\boldsymbol{s}|})\boldsymbol{V}^T, \boldsymbol{U}\text{diag}(\sqrt{|\boldsymbol{s}|})) + \lambda_o \sum_{l=1}^{D} L_o(\boldsymbol{U}_l, \boldsymbol{V}_l) + \lambda_s \sum_{l=1}^{D} L_s(\boldsymbol{s}_l). \quad (3)$$

Here L_T is the training loss computed on the model with decomposed layers. L_o denotes the orthogonality loss provided in Equation (1), which is calculated on the singular vector matrices $\boldsymbol{U}_l$ and $\boldsymbol{V}_l$ of layer l and added up over all D layers. L_s is the sparsity-inducing regularization loss, applying to the vector of singular values $\boldsymbol{s}_l$ of each layer. We explore the use of both the L^1 regularizer and the L^H regularizer (Equation (2)) as L_s in this work. λ_s and λ_o are the decay parameters for the sparsity-inducing regularization loss and the orthogonality loss respectively, which are hyperparameters of the proposed training process. λ_o can be chosen as a large positive number to enforce the orthogonality of singular vectors, and λ_s can be modified to explore the tradeoff between accuracy and FLOPs of the achieved low-rank model.

As shown in Figure 1, the low-rank decomposed network will be achieved through a three-stage process of *full-rank SVD training*, *singular value pruning* and *low-rank finetuning*. First we train a full-rank decomposed network using the objective function in Equation (3). Training at full rank enables the decomposed model to easily reach the performance of the original model, as there is no capacity loss during the full-rank decomposition. With the help of the sparsity-inducing regularizer, most of the singular values will be close to zero after the full-rank training process. Inspired by [35]'s work, we prune the singular values using an energy based threshold. For each layer we find a set $\mathbb{K}$ with the largest number of singular values subject to:

$$\sum_{j \in \mathbb{K}} s_j^2 \leq e \sum_{i=1}^{r} s_i^2, \quad (4)$$

where $e \in [0, 1]$ is a predefined energy threshold. We use the same threshold for all the layers in our experiments. When e is small enough, the singular values in set $\mathbb{K}$ and the corresponding singular vectors can be removed safely with negligible performance loss. The pruning step will dramatically reduce the rank of the decomposed layers. For a convolution layer with kernel $\mathbf{K} \in \mathbb{R}^{n \times c \times w \times h}$, if we can reduce the rank of the decomposed layers to r, the number of FLOPs for the convolution will be reduced by $\frac{(n+chw)r}{nchw}$ or $\frac{(nw+ch)r}{nchw}$ when channel-wise or spatial-wise decomposition is applied, respectively. The resulted low-rank model will then be finetuned with λ_s set to zero for further performance recovery.

Table 1. Comparison of top 1 accuracy on CIFAR-10 of ResNet models after full-rank SVD training, with or without orthogonality loss. [-Ch] means using channel-wise decomposition and [-Sp] means using spatial-wise decomposition.

Model	λ_o	Accuracy (%)
ResNet-56	N/A	93.14
ResNet-56-Ch	1.0	93.28
ResNet-56-Ch	0.0	91.28
ResNet-56-Sp	1.0	93.36
ResNet-56-Sp	0.0	90.70
Model	λ_o	Accuracy (%)
ResNet-110	N/A	93.62
ResNet-110-Ch	1.0	93.58
ResNet-110-Ch	0.0	91.83
ResNet-110-Sp	1.0	93.93
ResNet-110-Sp	0.0	91.86

4. Experiment Results

In this section, we first perform ablation studies on the effectiveness of each part of our training procedure using ResNet models [9] on the CIFAR-10 dataset [17]. We then apply the proposed decomposed training method on various DNN models on the CIFAR-10 dataset and the ImageNet ILSVRC-2012 dataset [28]. The training hyperparameters for these models can be found in Appendix A. Different hyperparameters are used to explore the accuracy-FLOPs trade-off induced by the proposed method. Our results constantly stay above the Pareto frontier of previous works.

4.1. Importance of the orthogonal constraints

Here we demonstrate the importance of adding the singular value orthogonality loss to the decomposed training process. We separately train two decomposed model with the same optimizer and hyperparameters, one with the orthogonality loss of $\lambda_o = 1.0$ and the other with $\lambda_o = 0$. No sparsity-inducing regularizer is applied to the singular values in this set of experiments. The experiments are conducted on ResNet-56 and ResNet-110 models, both trained under channel-wise decomposition and spatial-wise decomposition. The CIFAR-10 dataset is used for training and testing. As shown in Table 1, the orthogonality loss enables the decomposed model to achieve similar or even better accuracy comparing to that of the original full model. On the contrary, training the decomposed model without the orthogonality loss will cause around 2% accuracy loss.

4.2. Comparison of decomposition methods

As mentioned in Section 3.1, we mainly consider the channel-wise and the spatial-wise decomposition method

in this work. In this section, we compare the accuracy-#FLOPs tradeoff tendency of the channel-wise decomposition and the spatial-wise decomposition. The tradeoff tendency of both decomposition methods are explored by training the decomposed model with Hoyer regularizer of different strength (λ_s in Equation (3)) on the singular values. The results are shown in Figure 2. For shallower networks like ResNet-20 or ResNet-32 models, the spatial-wise decomposition shows a large advantage comparing to the channel-wise decomposition in the experiments, achieving significantly higher compression rate at similar accuracy. However, with a deeper network like ResNet-56 or ResNet-110, these two decomposition methods perform similarly. As discussed in Section 3.1, spatial-wise decomposition can utilize both spatial-wise redundancy and channel-wise redundancy, while the channel-wise decomposition utilizes channel-wise redundancy only. The observations in this set of experiments indicate that as DNN models get deeper, the channel-wise redundancy will become a dominant factor comparing to the spatial-wise redundancy. This corresponds to the fact that deeper layers in modern DNN typically have significantly more channels than shallower layers, resulting in significant channel-wise redundancy.

4.3. Comparison of sparsity-inducing regularizers

Under the same model decomposition scheme, the main factor related to the final compression rate and the performance of the compressed model would be the choice of sparsity-inducing regularizers for the singular values. As mentioned in Section 3.3, we mainly consider the use of the L^1 and the Hoyer regularizer in the proposed training scheme. In this section, we use spatial-wise decomposition setting to compare the effect of the L^1 regularizer and the Hoyer regularizer. A controlled group is also trained with no sparsity-inducing regularizer applied during the SVD training. The accuracy-#FLOPs tradeoff is explored by changing the regularization strength and singular value pruning threshold. All other hyperparameters are kept the same during SVD training and fintuning process for all models. Results are shown in Figure 3. The tradeoff tendency of the L^1 regularizer constantly demonstrates a larger slope than that of the Hoyer regularizer. Under low accuracy loss, the Hoyer regularizer achieves a higher compression rate comparing to that of the L^1 regularizer. However, if we are aiming for extremely high compression rate while allowing higher accuracy loss, the L^1 regularizer can have a better performance. One possible reason for the difference in tendency is that the L^1 regularizer will make all the singular values small through the training process, while the Hoyer regularizer will maintain the total energy of the singular values during the training, focusing more energy in larger singular values. Therefore more singular values can be removed from the decomposed model trained with the Hoyer regularizer without significantly hurting the performance of the model, resulting in higher compression rate at low accuracy loss. But it would be harder to keep most of the energy in a tiny amount of singular values than simply making everything closer to zero, therefore the L^1 regularizer may perform better in the case of extremely high speedup. Comparing to the controlled group with no sparsity-inducing regularization, both the L^1 regularizer and the Hoyer regularizer can achieve higher accuracy under similar compression rate, especially at high compression rate where the accuracy gap between with or without sparsity-inducing regularizer is more significant. Therefore applying sparsity-inducing regularizer on singular values is important for reaching a high performance low-rank model, as the weight will not naturally reach low-rank in training.

4.4. Effectiveness of the overall training procedure

To show the "full-rank SVD training, singular value pruning and low-rank finetuning" training framework proposed in Section 3.4 is essential for reaching a high performance low-rank model, we take the model architecture of the low-rank models achieved from the proposed training procedure, reinitialize all the weights with random values, and train the low-rank model from scratch. For fair comparison, the reinitialized low-rank model is trained using the same training objective and hyperparameter choices as the low-rank finetuning step in our framework. As shown in Table 2, with the same architecture and training process, training the low-rank model from scratch leads to around 2% testing accuracy loss comparing to the accuracy achieved by the proposed training procedure. This result correspond to the fact that low-rank models are harder to train from scratch due to their low capacity [35]. On the other hand, the full-rank SVD training step in our proposed framework provide sufficient capacity for the model to reach a high performance. Such high performance can still be preserved after singular value pruning, as the singular values are already sparse after the SVD training process.

4.5. Comparing with previous works

We apply the proposed SVD training framework on the ResNet-20, ResNet-32, ResNet-56 and ResNet-110 models on the CIFAR-10 dataset as well as the ResNet-18 and ResNet-50 model on the ImageNet ILSVRC-2012 dataset to compare the accuracy-#FLOPs tradeoff with previous methods. Here we mainly compare our method with state-of-the-art low-rank decomposition methods including Jaderberg et al. [15], Zhang et al. [39], TRP [35] and C-SGD [5], as well as recent filter pruning methods like NISP [37], SFP [10] and CNN-FCF [20]. The results of different models are shown in Figure 4. As analyzed in Section 4.3, the spatial-wise decomposition methods achieves significantly higher compression rate than the channel-wise

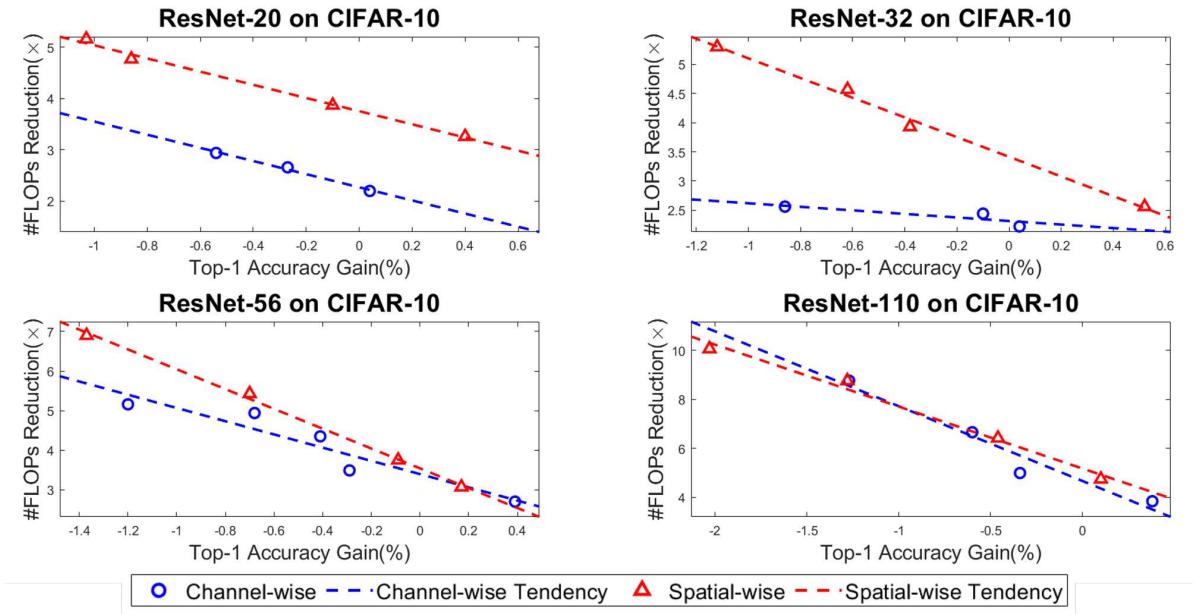

Figure 2. Effect of different decomposition methods. All models are achieved with Hoyer regularizer for singular value sparsity. Dash lines show the approximated tendency of the accuracy-compression tradeoff. See Appendix B Table 3 for detailed data.

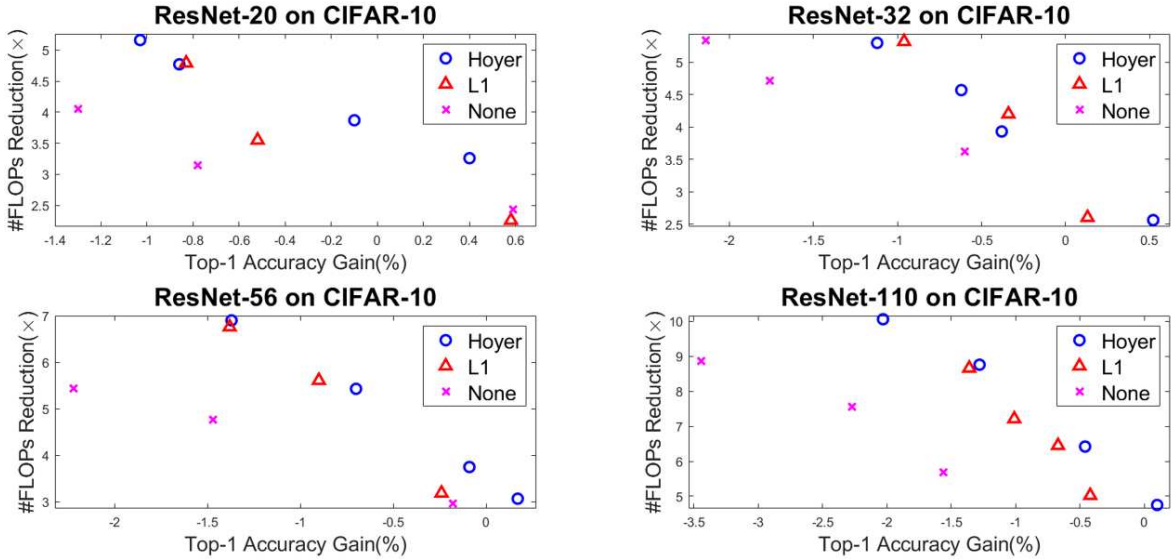

Figure 3. Effect of applying different sparsity-inducing regularizers during SVD training. All models are achieved with Spatial-wise decomposition. See Appendix B Table 3 for detailed data.

decomposition in shallower networks, while similar performance can be achieved when compressing a deeper model. Thus we compare the results of only the spatial-wise decomposition against previous works for ResNet-20 and ResNet-32. For other deeper networks, we report the results for both channel-wise and spatial-wise decomposition. As most of the previous works focus on compressing the model with a small accuracy loss, here we use the Hoyer regularizer for the singular values sparsity, as it can achieve a better compression rate than the L^1 norm under low accuracy loss (see Section 4.3). We use multiple strength for the Hoyer regularizer to explore the accuracy-#FLOPs tradeoff, in order to compare against previous works with different accuracy levels. As shown in Figure 4, our proposed method can constantly achieve higher FLOPs reduction with less accuracy loss comparing to previous methods on different models and datasets. These comparison results prove that the proposed SVD training and singular value pruning scheme can effectively compress modern deep neural networks through low-rank decomposition.

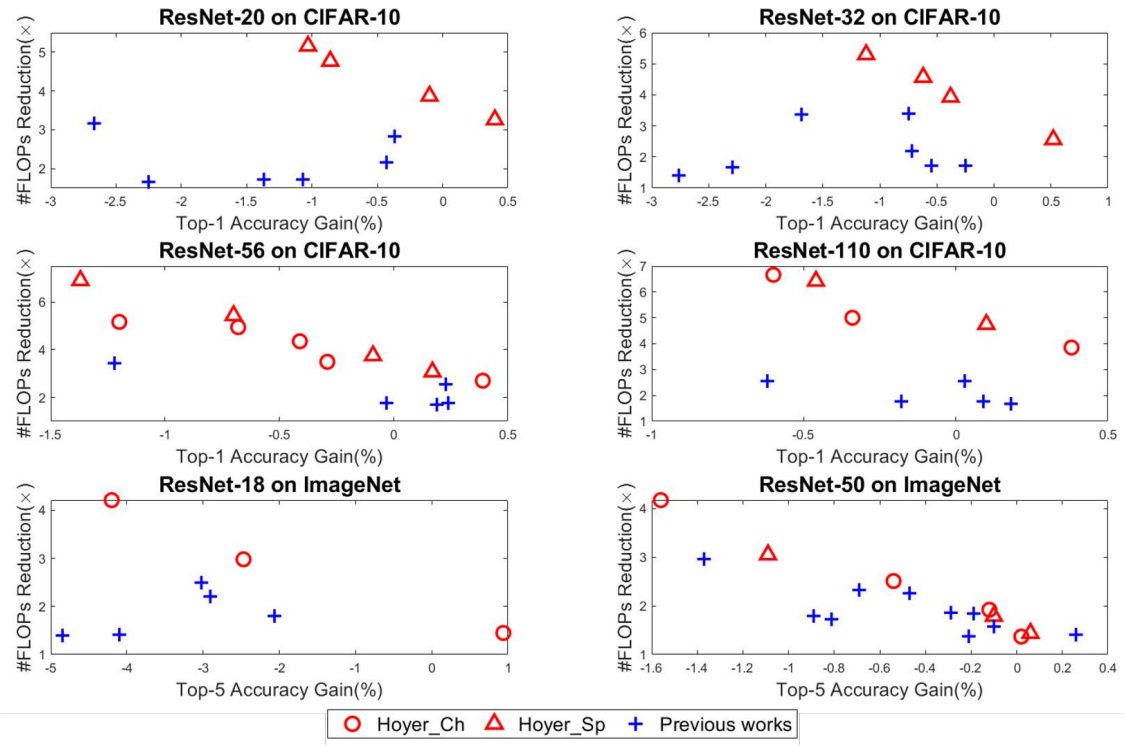

Figure 4. Comparison of accuracy-#FLOPs tradeoff against previous methods. Being closer to the top-right corner indicates a better tradeoff point. See Appendix B Table 3 - 5 for detailed results of our method, and Table 6 - 8 for all previous methods we compare to.

Table 2. Comparison of top 1 accuracy of the achieved low-rank ResNet models on CIFAR-10 vs. training the same model architecture from scratch. All low-rank models reported here are achieved with spatial-wise decomposition and Hoyer regularizer.

Base Model	Training method	Accuracy (%)
ResNet-20	Our method	91.39
Speed Up: 3.26×	From scratch	89.43
ResNet-32	Our method	91.76
Speed Up: 3.93×	From scratch	90.55
ResNet-56	Our method	93.27
Speed Up: 3.75×	From scratch	91.55
ResNet-110	Our method	93.47
Speed Up: 6.42×	From scratch	91.03

5. Conclusion

In this work, we propose the SVD training framework, which incorporates the full-rank decomposed training, singular value pruning and low-rank finetuning to reach low-rank DNNs with minor accuracy loss. We decompose each DNN layer to its full-rank SVD form before the training and directly train with the decomposed singular vectors and singular values, so we can keep an explicit measure of layers' ranks without performing the SVD on each step. Orthogonality regularizers are applied to the singular vectors during the training to keep the decomposed layers in a valid SVD form. And sparsity-inducing regularizers are applied to the singular values to explicitly induce low-rank layers.

Thorough experiments are done to analyse each proposed technique. We demonstrate that the orthogonality regularization on singular vectors is crucial to the performance of the decomposed training process. For decomposition methods, we find that the spatial-wise method performs better than channel-wise in shallower networks while the performances are similar for deeper models. For the sparsity-inducing regularizer, we show that higher compression rate can be achieved by Hoyer regularizer comparing to that of the L^1 regularizer under low accuracy loss. Our training framework is justified as training the low-rank model from scratch cannot reach the same accuracy achieved by our method. We further apply the proposed method to various depth of ResNet models on both CIFAR-10 and ImageNet dataset, where we find our accuracy-#FLOPs tradeoff constantly stays above the Pareto frontier of previous methods, including both factorization and structural pruning methods. These results prove that this work provides an effective way for learning low-rank deep neural networks.

Acknowledgments

This work was supported in part by NSF-1910299, NSF-1822085, DOE DE-SC0018064, and NSF IUCRC-1725456, as well as supports from Ergomotion, Inc.

References

[1] Jose M Alvarez and Mathieu Salzmann. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*, pages 856–867, 2017. 2

[2] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016. 1

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 1

[4] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014. 2

[5] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4943–4953, 2019. 1, 2, 6, 11

[6] M Giles. An extended collection of matrix derivative results for forward and reverse mode algorithmic differentiation. an extended version of a paper that appeared in the proceedings of ad2008. In *the 5th International Conference on Automatic Differentiation*, 2008. 2

[7] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 1

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 5

[10] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018. 6, 11

[11] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004. 4

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 1

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2

[15] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014. 1, 2, 3, 6, 11

[16] Dilip Krishnan, Terence Tay, and Rob Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR 2011*, pages 233–240. IEEE, 2011. 4

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5, 11

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[19] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014. 2

[20] Tuanhui Li, Baoyuan Wu, Yujiu Yang, Yanbo Fan, Yong Zhang, and Wei Liu. Compressing convolutional neural networks via factorized convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3977–3986, 2019. 1, 2, 6, 11

[21] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015. 1, 4

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[23] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 722–737, 2018. 1

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[25] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 1

[26] Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. Acdc: A structured efficient linear layer. *arXiv preprint arXiv:1511.05946*, 2015. 2

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 5, 11

[29] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015. 2, 3, 4

[30] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 4

[31] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019. 1

[32] Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. Learning intrinsic sparse structures within long short-term memory. *arXiv preprint arXiv:1709.05027*, 2017. 1

[33] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016. 1, 4

[34] Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating filters for faster deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 658–666, 2017. 2

[35] Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. Trained rank pruning for efficient deep neural networks. *arXiv preprint arXiv:1812.02402*, 2018. 1, 2, 3, 4, 5, 6, 11

[36] Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015. 1

[37] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018. 6, 11

[38] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018. 1

[39] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2015. 1, 2, 3, 6, 11